BASICによる代数計算の数式処理用のサブルーチン・パッケージ(数式処理と数学研究への応用)

AUTHOR(S):

増田, 真郎

CITATION:

増田, 真郎. BASICによる代数計算の数式処理用のサブルーチン・パッケージ(数式処理と数学研究への応用). 数理解析研究所講究録 1985, 551: 136-144

ISSUE DATE:

1985-02

URL:

http://hdl.handle.net/2433/98894

数理解析研究所講究録
551巻 1985年 136-144

BASICによる代数計算の数式処理用のサブルーチン・パッケージ

東京電機大理工　増田真郎（Masao Masuda）

数式処理の技法は，高度の科学計算のみならず，数学の初等・中等教育の段階においても不可欠のものであろう．実際パーソナル・コンピュータが急速に普及している割に教育現場での使用が遅れているのは，数式処理の技法の啓蒙が余り行われておらず，数値計算志向のコンピュータ利用技術のみでは，学校教育とコンピュータとの間隙が大きすぎることに起因するものと思われる．

パーソナル・コンピュータ用の数式処理システムとしてはすでにμ-MATHが知られているが，数式処理の啓蒙普及のためには，大衆的なBASICを利用できる方が効果的である．

したがって，使用言語をBASICに限定し，また対象を代数計算に止めて，数式処理用のサブルーチン・パッケージを作成することを試みた．

計算の規模は手計算を若干上廻る程度とし，処理時間に対

する要求も控え目としたが，その反面パーソナル・コンピュータを電卓的に用いて，数式等の計算を可能にすることも一つの目標とした．

数式処理における基本問題は，数式処理とは何かという点であろう．数式処理とは，数式の形で入力し，結果が数式として出力されることであると述べられていることもあるが，むしろ“数式を数式のままで取り扱うこと”と定義する方がニュアンスとしては適当であるように思われる．

前者の表現では，積分計算の場合や記号処理向き言語を用いる場合はともかくとして，代数計算をBASICのような数値計算向き言語で実行する場合に，安易な考え方に導きやすい．

たとえば多項式計算の場合，数式のまま入・出力するためには，入・出力が文字列であることは当然であるが，入力後すぐにその係数を数値配列変数でとろうとするのが常である．もちろんこの方法でも多項式の加・減・乗算や，商と余りを求める割り算を実行し，結果を式の形にまとめて文字列として出力することは容易である．これでも見かけ上は確かに数式処理となっている．

しかし，係数が分数形式で表示された有理数，さらにその代数拡大の元となるにつれて，係数を保持する配列変数が多

次元となり，取り扱うことはできても，事は急速に面倒になる．有理式・2変数以上の多項式などの取り扱いにいたっては，それはほぼ困難の領域に入るに違いない．

結局このような方法は，入・出力のみが数式である数式処理である点に問題があるのであろう．

一般に，代数系Sにおける計算を，われわれは

(1) 文字列としてのSの元を読む，

(2) 必要な情報を取り出す，

(3) 計算，

(4) 結果として与えられた情報を整理する，

(5) 文字列としてのSの元を書く

の順に行っている．

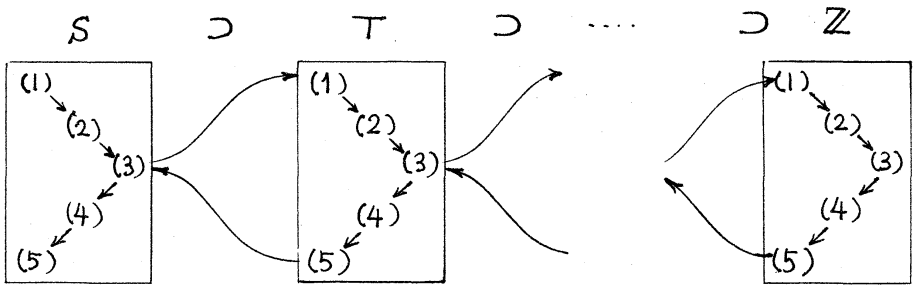
Sの演算が部分代数系$T \subset S$における演算で定義されているときは，(3)の部分はTにおける計算である．

この関係は連鎖状であり，たとえば1変数有理係数有理関数体の四則計算は，部分代数系の連鎖

$$\mathbb{Q}(X) \supset \mathbb{Q}[X] \supset \mathbb{Q} \supset \mathbb{Z}$$

において，次々に下位の代数系との計算の受け渡しによって実行されている．

したがって，$\mathbb{Q} \subset S$ または $\mathbb{F}_p \subset S$ の場合，最終的な計算は$\mathbb{Z}$における計算に帰せられる．



これらの各代数系における計算をそれぞれ行う場合には，数式処理の最低条件として入・出力 (1)，(5) は文字列で行われるはずである．したがって，これらを連鎖的に結合して計算するのであれば，計算 (3) における直下位の代数系とのデータの受け渡しを文字列で行うのが自然であり，そのためには (2)，(4) も文字列処理である方がよい．

実際，すでに習慣となっているためほとんど意識をしていないが，手計算の場合われわれはまさにこの通りのことを行っており，$\mathbb{Z}$ における最終的な計算 —— 10進1ないし2桁の計算 —— を除いては，各代数系において (1)～(5) はすべて文字列処理である．

したがって，この最終的な整数の計算を BASIC の演算機能にゆだねることにして，ほかはすべて文字列処理として各代数系の演算用のサブルーチンを構築していくと，つねに数式を数式のままで取り扱っている数式処理システムが BASIC によって作られるはずである．幸に，BASIC は文字列処理に

関してほぼ十分な機能をもっているので，この考え方の実現は容易である．

代数系Sの演算用サブルーチンでは，たとえば2項演算の場合，Sの2元と演算の種類（それを＋，－，＊，／などの文字列で表わす）をそれぞれ文字型変数 ＊X，＊Y，＊OPで受けて，演算結果を文字型変数 ＊Zに入れて返すようにする．ここに＊は各代数系に対して固有に定める．

実際に作成されたサブルーチンでは，これらは次のように設定されている（試行錯誤の結果作成されたので，統一性にやや欠けている）．

$\mathbb{Z}$	（有理整数環）	X, Y, ROP → Z	(8000)
$\mathbb{Q}$	（有理数体）	RX, RY, SOP → RZ	(6800)
$\mathbb{F}_p$	（有限体 $\mathbb{Z}/(p)$）		
$\mathbb{Q}(\sqrt{m})$	（2次体）	WX, WY, WOP → WZ	(5500)
$\mathbb{F}_p(\alpha)$	（$\mathbb{F}_p$の有限次拡大体）		
K[X]	（体K上の1変数多項式環）	PX, PY, POP → PZ	(4000)
K(X)	（体K上の1変数有理関数体）	QX, QY, QOP → QZ	(3200)
K[X, Y]	（体K上の2変数多項式環）	DX, DY, DOP → DZ	(1900)
$V^n(K)$	（体K上のn次元ベクトル空間）	VX, VY, UOP → VZ	(1800)
$M_{m,n}(K)$	（体K上の(m,n)行列全体）	TX, TY, XOP → TZ	(2800)

BASICによっては，ラベルの使用できないものもあるので，

サブルーチン名はすべてその最初の文番号とし，同じ文字型変数で受け渡しが行われるサブルーチンは同一の文番号から始め，また連鎖状に結合されるはずのサブルーチンの間では文番号の重複がないようにする．前記の（ ）内の数字が実際に作成されたサブルーチンの開始文番号である．

この結果，たとえばサブルーチンの結合（以下代数系Sの演算用のサブルーチンを同じ文字Sで表わす）

（主プログラム）—— K[X] —— Q ——Z　　(1)

によって有理係数多項式の計算が可能となるが，同じ主プログラムで

（主プログラム）—— K[X] —— $\mathbb{F}_p$ —— Z　　(2)

のように結合することによって，有限体$\mathbb{F}_p$上の多項式の同じ計算が実行できる．

またこの主プログラムおよびサブルーチンK[X]で，

RX，RY，RZ，ROP　をそれぞれ　WX，WY，WZ，WOP

に置き換え，GOSUB文の文番号を変更すれば，

（主プログラム）—— K[X] —— $\mathbb{Q}(\sqrt{m})$ — Q —— Z　　(3)

（主プログラム）—— K[X] —— $\mathbb{F}_p(\alpha)$ —— $\mathbb{F}_p$ — Z　　(4)

という結合によって，2次体$\mathbb{Q}(\sqrt{m})$上，有限体$\mathbb{F}_p(\alpha)$上の多項式の同じ計算が実行できる．

実際，たとえば主プログラムの中で，文字型変数PF，PG

に代入されている多項式の和 PH は，サブルーチン K[X] の開始文番号が 4000 であるから，

PX = PF : PY = PG : POP = "+" : GOSUB 4000 : PH = PZ

によって計算される．サブルーチン K[X] の中で，多項式 PX, PY の I 次，J 次の項がそれぞれ PX(I), PY(J) であるとき，その積 PZ(L) を求めるのであれば，(1), (2) の場合には

RX = PX(I) : RY = PY(I) : SOP = "*" : GOSUB 6800 : PZ(L) = RZ

とすればよく，また (3), (4) の場合は

WX = PX(I) : WY = PY(I) : WOP = "*" : GOSUB 5500 : PZ(L) = WZ

とすればよい．

もちろん (3), (4) によって (1), (2) の場合の計算が可能であるが，計算時間を短縮するためには区別する方が効率的である．

四則演算のほか，もちろん $\mathbb{Z}$ や K[X] に対しては，商と余りを求めるサブルーチンも用意する．たとえば K[X] においては

PXX と PYY で割った商を PQ，余りを PR に入れて返すように作ると，PF, PG の最大公約式 PD を求める互除法の計算は

10 PXX = PF : PYY = PG

```
20  GOSUB 4400 :  IF PR="0" THEN 50
30  PZ=PR : GOSUB 5050
40  PXX=PYY :  PYY=PZ : GOTO 20
50  PD=PYY
```

とすればよい．ここに4400は商と余りを求めるサブルーチンの開始文番号，5050は多項式PZをモニック多項式PZになおすサブルーチンの開始文番号である．計算の途中，いつPXX，PYYを取り出しても，それは文字列としての数式を保持している．

つねに計算の対象が文字型変数に代入されているから，スタックとして文字型配列変数S(1)，S(2)，…を用意すれば，いずれの代数系の場合も各S(K)に1つずつその元を格納できる．したがって，この代数系の元や演算記号等をキーボードから次々に入力することによって，逆ポーランド法による連続計算が実行できる．

(m, n)行列の場合にも，行列を1次元行ベクトルの形に直して，これを文字列として1つの文字型変数で取り扱うことにすれば，全く同様にして連続計算が可能である．

これらのサブルーチンにおいて，配列変数の占める記憶領域はほとんど問題にならない．むしろBASICの場合，1つの文字型変数に代入できる文字列の長さが255字以内であ

るため，計算規模はそれによって制限される．

しかし，文字列の分解・結合を繰り返している割にはそれほど計算時間は遅くならず，パーソナル・コンピュータとして許容できる範囲に止っているように思われる．そして何よりも主プログラムの作成が容易であり，たとえばHenselの補題にもとづく計算を実行するプログラムも，不自然でない程度にマルチ・ステートメントを許すとして，60行ほどで書くことができる．

作成の動機であった初等・中等教育に対する配慮からはそれたが，ここでとった考え方はそのまま初等的問題にも適用できる．

一方，文字列処理に関してBASICと類似しているPL/Iによってこれらのサブルーチンを書き直せば，計算規模は十分に拡大でき，研究目的にも耐えられるような大型コンピュータ上の代数計算用の数式処理サブルーチン・パッケージがえられるであろう．